Our Ref. No. 042390.P8110
Express Mail No.: EL851136755US

UNITED STATES PATENT APPLICATION

FOR

FILE CHECKING USING REMOTE SIGNING AUTHORITY VIA A NETWORK

INVENTORS:

Carl M. Ellison
Roger A. Golliver
Howard C. Herbert
Derrick C. Lin
Francis X. McKeen
Gil Neiger
Ken Reneris
James A. Sutton
Shreekant S. Thakkar

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

BACKGROUND

1. Field

This invention relates to microprocessors. In particular, the invention relates to processor security.

2. General Background

Advances in microprocessor and communication technologies have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (E-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while modern microprocessor systems provide users convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable for unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Computer security, therefore, is becoming more and more important to protect the integrity of the computer systems and increase the trust of users.

Threats caused by unscrupulous attacks may occur in a number of forms. For instance, an invasive remote-launched attack by hackers may disrupt the normal operation of a system connected to thousands or even millions of users. A virus program may corrupt code and/or data operating on a single-user platform or may propagate itself to other platforms when connected to a network. Although anti-virus programs have been developed to scan, detect and eliminate known viruses, a large performance penalty would be incurred if an anti-virus program is required to examine every file before it can be opened.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1 is an exemplary embodiment of a network configured for that a first embodiment of the invention can be practiced.

Figure 2 is an exemplary embodiment of a platform employed in a network as shown in Figure 1.

Figure 3 is an exemplary embodiment of a verification scheme performed by the requesting platform to verify that the signatory failed to detect an abnormality in the uploaded file.

Figure 4 is an exemplary embodiment of a network configured for that a second embodiment of the invention can be practiced.

Figure 5 is an exemplary embodiment of a verification scheme performed by the requesting platform to verify that the signatory has been authorized to perform the file checking scheme and failed to detect an abnormality in the uploaded file.

Figure 6 is an exemplary embodiment of a network configured for that a third embodiment of the invention can be practiced.

Figure 7 is an exemplary embodiment of a file checking mechanism in accordance with one embodiment of the invention.

Figure 8 is an exemplary embodiment of a flowchart of one embodiment of a remote file checking mechanism.

DESCRIPTION

The invention relates in general to a method and apparatus to check file integrity remotely. In one embodiment, a file is sent from a platform to a signatory via a network. The signatory checks the file and a digital signature chain is returned to the platform upon verifying the integrity of the file. As an alternative embodiment, the file checking operation is performed internally within the platform.

Within the platform, the file is accessed based on a verified digital signature chain. The file is not opened if (1) no digital signature chain is associated with the file, (2) the digital signature chain is provided by an unauthorized signatory, or (3) the digital signature chain indicates an unacceptable file integrity upon verification. The file may be opened if the verified digital signature chain indicates acceptable file integrity.

Herein, terminology is used to discuss certain features of the present invention. For example, a "platform" may generally be considered as hardware equipment and/or software that process information. Some illustrative examples of a platform include a computer (e.g., desktop, a laptop, a hand-held, a server, a workstation, etc.), communication device (e.g., router, bridge, brouter, etc.), a wireless telephone handset, a television set-top box, and the like. A "file" is generally considered herein as a collection of information in a selected format. Various types of files include code (e.g., source, object, executable, applets, operating systems, etc.), a digital document (e.g., word processing, spreadsheet, etc.), an electronic mail (e-mail) message and the like. "Information" includes data, address and/or control.

With respect to cryptography related terminology, a "key" is an encoding and/or decoding parameter. The term "signatory" is defined as a manufacturer, a trade association, a governmental entity, a bank, a particular department of a company (e.g., security or the information technology "IT" department or any other entity or person in a position of trust) and/or a platform controlled by the signatory. A "digital signature chain" includes an ordered sequence of digital signatures and/or certificates arranged for authorization purposes, where a certificate may be used to authenticate the authority of a signatory of a corresponding digital signature.

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention.

A. ARCHITECTURE OVERVIEW: FILE CHECKER IMPLEMENTED IN SIGNATORY DIRECTLY IN COMMUNICATIONS WITH THE REQUESTING PLATFORM

Referring to Figure 1, an exemplary embodiment of a network 10 adapted to perform an embodiment of the invention is shown. The network 10 includes a subnetwork 20, a wide area network (WAN) 60, and/or a remote site 70. A file checking mechanism (referred to as a "file checker") is hardware and/or software configured to check file integrity, detect virus infection, detect intrusion or any combination thereof. The file checker may be employed within a platform of the subnetwork 20 or coupled to a local area network (LAN) connection of the subnetwork 20 as described below. Alternatively, the file checker may be employed within the remote site 70 in communication with the WAN 60.

The subnetwork 20 represents a local area network (LAN) in a network system. The subnetwork 20 includes a network server 25, a LAN connection 30, platforms 35_1-35_M ("M" being a whole number, $M \geq 1$) and/or a local signatory 40. The subnetwork 20 is typically an intranet or a group within an organization. The subnetwork 20 connects all users of the platforms 35_1-35_M and the signatory of a signatory 40 in the group together. When used in association with a signatory, the term "local" generally means that the signatory 40 is normally closer in physical proximity to the platforms 35_1-35_M and directly connected to the subnetwork 20, and is used to distinguish from a remote signatory as discussed later.

The subnetwork 20 allows these users to participate in group activities such as conferencing, meeting, information exchange, document downloading, and resource sharing. In particular, the subnetwork 20 allows one of the platforms 35_1-35_M (e.g., the platform 35_1) to request the signatory 40 to analysis the integrity of an uploaded file and to produce a digital signature as an output if the integrity of the uploaded file is verified. The network server 25 provides users of the LAN accesses to the WAN 60.

Referring now to Figure 2, an exemplary embodiment of any platform 35_1-35_M is shown. For instance, platform 35_1 comprises a processor 110, a host bus 120, a first control unit 130, and a system memory 140. As an option, the first platform 20 further comprises a second control unit 150, a non-volatile memory or system flash 160, a mass storage device 170, input/output devices 175, a token bus 180, a motherboard (MB) token 182, a reader 184, and other types of token(s) 186. The first control unit 130 may be integrated into a chipset that integrates multiple functionalities including memory control. Similarly, the second control unit 150 may also be integrated into a chipset together or separate from the first control unit

130 to perform input/output (I/O) functions. For clarity, not all of the peripheral buses are shown. It is contemplated that the platform 35_1 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with an Intel Architecture (IA) processor, such as the PENTIUM® series, the IA-32™ and the IA-64™.

In one embodiment, the platform 35_1 can be a single processor system, such as a desktop computer, which has only one main central processing unit, e.g. processor 110. In other embodiments, the platform 35_1 can include multiple processors, e.g. processors 110, 110a, 110b, etc., as optionally shown by dashed lines. Thus, the platform 35_1 can be a multi-processor system having any number of processors. For example, the multi-processor system can operate as part of a server or workstation environment. It will be appreciated by those skilled in the art that the basic description and operation of processor 110 applies to the other processors 110a and 110b as well as any number of other processors that may be utilized in the multi-processor system according to one embodiment of the invention.

The processor 110 may also have multiple logical processors. A logical processor, sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according

to some partitioning policy. A multi-threaded processor is a processor having multiple threads or multiple logical processors. Thus, a multi-processor system may have multiple multi-threaded processors.

The host bus 120 provides interface signals to allow the processor(s) 110, 110a, and/or 110b to communicate with other processors or devices, e.g., the first control unit 130. Herein, the first control unit 130 provides control and configuration of memory and I/O devices such as the system memory 140 or the second control unit 150. The first control unit 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the first control unit 130 may include memory range registers (e.g., base and length registers) to represent an amount of access protected area in the system memory 140.

The system memory 140 stores files such as code and/or data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). In one embodiment, system memory 140 may be partitioned into an accessible area 141 and an isolated area 142. Access to the isolated area 142 is restricted and is enforced by the processor 110 and/or the first control unit 130.

The second control unit 150 includes a digest memory 154, a cryptographic key storage 155, and a token bus interface 159. The digest memory 154, typically implemented in RAM, stores one or more digests (e.g., hash values) of various files. The cryptographic key storage 155 holds one or more keys that are unique for the platform of the platform 35_1. In one embodiment, the cryptographic key storage 155 includes internal fuses that are programmed at manufacturing. Alternatively, the cryptographic key storage 155 may also be

created with a random number generator and a strap of a pin. The token bus interface 159 interfaces to the token bus 180.

Certain secondary devices are in communication with and, in some instances, under control of the second control unit 150. For example, the internal memory 160 stores information in a non-volatile manner. Typically, the internal memory 160 is implemented with flash memory. The mass storage device 170 stores archive information (e.g., files) on machine-readable media and provides a mechanism to read information from the machine-readable media. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard drive 176, and any other magnetic or optic storage devices.

When implemented in software, the elements of the present invention are code segments performing necessary tasks. The program or code segments can be stored in machine-readable medium or embodied in a signal propagating over a transmission medium. The "machine-readable medium" may include any medium that can store or transfer information. Examples of the machine-readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optic medium, a radio frequency (RF) link, etc. Examples of the "transmission medium" include electrical conduits (wire, bus traces, etc.), optical fiber(s), air, and the like. The code segments may be downloaded via computer networks such as the Internet, Intranet, etc.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include controller for input devices (e.g., keyboard,

mouse, trackball, pointing device), media card (e.g., audio, video, graphics), network card, and any other peripheral controllers.

The token bus 180 provides an interface between the second control unit 150 and various tokens in the platform. A token is a device that performs dedicated input/output functions with security functionalities. A token has characteristics similar to a smart card, including one or more keys and the ability to sign data. Examples of tokens connected to the token bus 180 include a motherboard token 182, a token reader 184, and other portable tokens 186 (e.g., smart card, biometric identifier, etc.).

Referring back to Figure 1, either the local signatory 40 or the remote signatory 80 (referred to generically as "signatory 40/80") is implemented with a file checker 45 to check the integrity of an uploaded file 50 provided by the requesting platform 35_1. In general, the file 50 contains code, data, or a combination thereof. The requesting platform 35_1 may acquire the file 50 through any number of ways. For example, the requesting platform 35_1 may receive the file 50 from another platform, either within the subnetwork 20 or in any other subnetwork. The requesting platform 35_1 may also acquire the file 50 via other media such as from a floppy diskette, a CD ROM, or by downloading a file attached to an e-mail or from a commercial site. After acquiring the file 50, the requesting platform 35_1 does not attempt to open it. Instead, the requesting platform 35_1 assumes the file 50 is bad until the file integrity has been verified by the signatory 40/80.

In particular, with respect to the first embodiment of the invention, the requesting platform 35_1 requests file checking by routing the file 50 to the signatory 40 as shown in Figure 1. Implemented with file checker 45, the

signatory 40 analyzes the uploaded file 50 to verify file integrity and has the authority or capability to issue a digital signature chain associated with the uploaded file 50. In one embodiment, the signatory 40 utilizes a platform employing the file checker 45 as shown above.

Herein, as one embodiment, the file checker 45 is typically either an anti-virus program, a virus detector, or an intrusion detector. The virus detector may be a commercial virus detector program or a specially designed virus detector. Examples of the file checker include MCAFEE® programs, NORTON® antivirus programs and the like.

The signatory 40 receives the file 50 via the LAN connection 30. It is noted that when the file 50 is sent via the network, either LAN or WAN, there is a chance of a security breach. The file 50 may be intercepted by an intruder monitoring the network traffic. In an intranet or group environment, this scenario is highly unlikely because the security of the network is tight. Over the WAN 60, however, the probability for security breach is higher and therefore this mechanism is more suitable for files without encryption requirements.

After receiving the file 50, the signatory 40 analyzes the file 50 and detects if there is any virus infection or intrusion. The signatory 40 then generates a digital signature chain 55 (e.g., a digital signature) that verifies the integrity of the file 50, and returns the digital signature chain 55 back to the requesting platform 35_1. When there are many files to be checked, there may be a need to identify which file the signatory 40 is associated with. The signatory 40, therefore, may contain a file identifier so that the requesting platform 35_1 can know which file the signatory 40 is associated with.Referring now to Figure 3, one exemplary embodiment of a verification scheme performed by the requesting platform 35_1 to

verify that the signatory 40 failed to detect an abnormality in the uploaded file 50 is shown. Upon receipt of the digital signature chain 55, namely a digital signature for clarity sake, the platform 35_1 recovers contents of the digital signature. The recovered contents of the digital signature include a digest 200 of the uploaded file. In addition, the file 50 undergoes a hash function 210 to produce a digest 220. If the digest 210 matches the recovered digest 200, the integrity of the file 50 has been verified and the platform 35_1 allows the file to be opened and/or executed.

Of course, verification scheme described above is for illustrative purposes only. Other verification schemes are possible. For example, the contents (e.g. an alphanumeric statement) may be recovered from the digital signature chain 55. The contents may indicate if the file integrity is acceptable, unacceptable or questionable, requiring human analysis.

B. ARCHITECTURE OVERVIEW: FILE CHECKER IMPLEMENTED IN REMOTE SIGNATORY INDIRECTLY IN COMMUNICATIONS WITH THE REQUESTING PLATFORM

Referring to Figure 4, an exemplary embodiment of a network 10 configured in accordance with a second embodiment of the invention is shown. As described above, the network 10 includes the subnetwork 20, the WAN 60 and the remote site 70. Herein, for this embodiment, the WAN 60 provides public accesses to other subnetworks or commercial sites. The WAN 60 may be the Internet, the world wide web (WWW), or any other wide area networks. The WAN 60 includes network switches/routers 65_1-65_L (when $_L \geq 1$). The network switches/ routers 65_1-65_L regulate and route traffic in the network 10. The network switches/ routers 65_1-65_L are linked by network-network interface (NNI)

links. The network switches/ routers may be asynchronous transfer mode (ATM) switches/ routers, or any other network switches or routers.

The remote site 70 provides services to the public or registered users. The remote site 70 includes a server 75 and a remote signatory 80. The server 75 provides connection to the WAN 60 to handle incoming and outgoing traffic. The remote signatory 80 is capable of digitally signing files received from other subnetworks such as the subnetwork 20. The remote signatory 80 also has the ability to check file integrity, detect virus infection, and intrusion. One example of the remote signatory 80 may include a website managed by McAfee.com Corporation or Symantec Corporation of Cupertino, California (NORTON® antivirus tools).

In particular, with respect to the second embodiment of the invention, the requesting platform 35_1 requests file checking remotely by routing the file 50 to the local signatory 40. In response, the local signatory 40 redirects the file 50 to the remote signatory 80. Implemented with file checker 45, the remote signatory 80 analyzes the uploaded file 50 to verify file integrity and has the authority or capability provided from the remote signatory 80 to issue a digital signature associated with the uploaded file 50. The remote signatory 80 may employ a platform running the file checker 45.

After receiving the file 50, the remote signatory 80 analyzes the file 50 and detects if there is any virus infection or intrusion. The remote signatory 80 then generates a digital signature 56 (e.g., a digital signature as shown) that verifies the integrity of the file 50, and returns the digital signature 56 back to the local signatory 40. In response to receiving the digital signature 56, the local signatory 40 provides the digital signature chain 55, including the digital signature 56 and

its accompanying digital certificate 57. The digital certificate 57 provides information to the platform 35_1 that the remote signatory 80 has been authorized by the local signatory 40 to analyze the uploaded file 50.

Alternatively, it is contemplated that the local signatory may be, in effect, implemented in connection with a firewall (e.g., an application gateway) that is configured to preclude transmission and reception of incoming information in certain situations. For instance, for incoming (or even outgoing) files (or email messages) without a corresponding digital signature chain, the local signatory 40 could preclude re-routing of the file to a targeted platform, which is coupled to the LAN, until one of two conditions exists. One condition is for the file checker 45 of the local signatory 40 to receive the file, verify its integrity, and issue a proper digital signature chain to accompany the file if its integrity is verified and acceptable. For files already with a digital signature chain, the local signatory 40 could preclude re-routing of the file to a targeted platform on the LAN unless the digital signature chain has been verified by the local signatory 40.Referring now to Figure 5, an exemplary embodiment of a verification scheme performed by the requesting platform 35_1 to verify that the remote signatory 80 has been authorized to perform the file checking scheme and failed to detect an abnormality in the uploaded file 50 is shown. Upon receipt of a digital signature chain 58, inclusive of the digital signature 56 and the digital certificate 57, the platform 35_1 recovers contents of the digital certificate 57 using a public key (PUK_L) 300 of the local signatory 40. The contents of the digital certificate include a public key (PUK_S) 310 of the remote signatory 80. PUK_S is used to recover a digest 320 of the uploaded file 50 contained in the digital signature 57. In addition, the file 50 undergoes a hash function 330 to produce a digest 340. If the digest 340 matches the recovered digest 320, the integrity of the file 50 has been verified and the

platform 35_1 allows the file to be opened and/or executed. Of course, other verification schemes inclusive of those described above may be used.

C. ARCHITECTURE OVERVIEW: FILE CHECKER IMPLEMENTED IN THE REQUESTING PLATFORM

In a third embodiment as shown in Figure 6, the requesting platform 35_1 is implemented with the file checker 45 to verify the file integrity. The file checker 45 interfaces to the file 50 to be checked. The requesting platform 35_1 may have acquired the file 50 from any number of ways. After acquiring the file 50, the requesting platform 35_1 does not attempt to open it and assumes the file 50 is bad until its integrity is verified by the file checker 45.

D. FILE CHECKER

The basic idea of the invention is to enforce a policy for checking file integrity against virus(es) or intrusion. According to this policy, an unknown file is not opened unless its file integrity is verified. An unknown file is a file that has just been created (e.g., a new file), or that has just been closed (e.g., a modified file). By refusing to open a file with a signature indicating unacceptable file integrity, or without a signature, the platform can be guaranteed that there will be no opportunities for virus to spread out infecting other files or elements.

The file checker 45 checks file integrity of files in a platform. The file checker 45 comprises a file analyzer 700 and a signature generator 710. The file analyzer 700 receives the original file 50 and produces a scanned file 720. The scanned file 720 is the original file 50 after performance of one or more scan operations.

In particular, the file analyzer 700 is a facility to perform scan operations on the original file 265 and return the scanned file 280. The scan operations include, but are not limited or restricted to a virus detection, an intrusion detection, a file integrity detection, or any appropriate program. The virus detection may be a commercial anti-virus program or virus scanner such as the MCAFEE® virus scanner, or an intrusion detector based on an expert system or an artificial immune system. The file analyzer 700 generates the scanning result 730 according to the result of the scan. The scanning result 730 may indicate that the original file 50 has an acceptable file integrity (e.g., virus free), an unacceptable file integrity (e.g., infected with virus), or a questionable integrity which may require in-person analysis of the file.

The signature generator 710 receives the scanned file 720 and optimally the result 730 (represented by dashed lines). Thereafter, the signature generator 710 produces a digital signature 740. The digital signature 740 may be part of the digital signature chain 55, described above.

It is further contemplated that the file checker 45 is optimally implemented with a time stamp indicator 750. The time stamp indicator 750 provides information regarding the recency of the scan operation. In one embodiment, the time stamp indicator 750 is one of a calendar time obtained from the platform.Figure 8 is a flowchart illustrating a process 800 for remote file checking according to one embodiment of the invention.

Initially, the process 800 determines if the file has a corresponding digital signature chain (Block 810). If so, the process 800 verifies the digital signature chain as described in block 860. Otherwise, the process 800 sends the file to the signatory via a network (Block 820). The signatory checks the file integrity

(Block 830). For instance, this can be done by performing a scan operation on the file using a file checker (e.g., a virus detector, an intrusion detector, etc.). Next, the signatory generates and sends a digital signature chain associated with the file indicating the result of the checking via the network (Blocks 840 and 850).

Next, the digital signature chain is verified and generates a verified signature or result (Block 860). Then, a determination is made if the verified signature indicates an acceptable file integrity (Block 870). If not, the files will not be opened or executed and a failure or fault condition is generated to notify the user (Blocks 880 and 890). The process is then terminated. However, if the verified signature indicates acceptable file integrity, the process proceeds to open or execute the file at the user's request (Block 885). The process is then terminated.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.